

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

8

§

8

2

8

8

2

32

22

JUN 04 2004

Technology Center 2100

By:

Amy Miller

(Appellant's Brief Page 1 of 21)
Curtis et al. – 09/578,750

REAL PARTIES IN INTEREST

The real party in interest in this appeal is the following party: IBM Corporation

RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such appeals or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-39

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims canceled: NONE
2. Claims withdrawn from consideration but not canceled: NONE
3. Claims pending: 1-39
4. Claims allowed: NONE
5. Claims rejected: 1-39

C. CLAIMS ON APPEAL

The claims on appeal are: 1-39

STATUS OF AMENDMENTS

There are no amendments after final rejection.



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

AF
2122
\$

In re application of: **Curtis et al.**

Serial No.: **09/578,750**

Filed: **May 25, 2000**

**For: Method of Applying an Update to
a Contained Collection of Program
and Data Files Based Upon Versions**

35525

PATENT TRADEMARK OFFICE
CUSTOMER NUMBER

§ Group Art Unit: **2122**
§
§ Examiner: **Kendall, Chuck O.**
§
§ Attorney Docket No.: **AUS9-2000-0214-US1**
§

§ Certificate of Mailing Under 37 C.F.R. § 1.8(a)
§ I hereby certify this correspondence is being deposited with the United
§ States Postal Service as First Class mail in an envelope addressed to:
§ Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on
May 28, 2004
By: Amy Miller
Amy Miller

RECEIVED

JUN 04 2004

Technology Center 2100

TRANSMITTAL DOCUMENT

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:
ENCLOSED HEREWITH:

- Appellant's Brief (in triplicate) (37 C.F.R. 1.192); and
- Our return postcard.

A fee of \$330.00 is required for filing an Appellant's Brief. Please charge this fee to IBM Corporation Deposit Account No. 09-0447. No additional fees are believed to be necessary. If, however, any additional fees are required, I authorize the Commissioner to charge these fees which may be required to IBM Corporation Deposit Account No. 09-0447. No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to Deposit Account No. 09-0447.

Respectfully submitted,

Duke W. Yee

Duke W. Yee

Registration No. 34,285

YEE & ASSOCIATES, P.C.

P.O. Box 802333

Dallas, Texas 75380

(972) 367-2001

ATTORNEY FOR APPLICANTS

SUMMARY OF INVENTION

The present invention provides a method, system, and apparatus for updating code of a software installer program. In a preferred embodiment a program, such as a patch, is provided to a plurality of versions of an install program, wherein the program is updated by an installation program and a plurality of versions of the installer program exist. See specification, page 12, lines 12-25; page 14, lines 7-18. Next, it is determined whether the version of the installer program is incorrect. See specification, page 14, lines 18-23. If the version is old, the installer program is updated from files in the patch. See specification, page 14, line 27, to page 15, line 4. The patch is then installed into the program using the updated installer program. See specification, page 15, lines 5-10.

ISSUES

The issues on appeal are as follows:

Whether claims 1-7, 10-16, 20-25, 28, 29, 33, 36, and 37 are unpatentable as being anticipated by *Smith et al.* (U.S. Patent No. 6,477,703) in view of Chamberlain (U.S. Patent No. 6,427,227).

Whether claims 8, 9, 17, 18, 26, 27, 30, 31, 34, 35, 38, and 39 are unpatentable as being obvious over *Smith et al.* in view of Chamberlain and further in view of *Forbes et al.* (U.S. Patent No. 6,381,742).

GROUPING OF CLAIMS

The claims on appeal do not stand or fall in a single group, but are grouped into the following groups for the reasons set forth in the Argument section below:

Claims 1, 2, 7, 10, 11, 16, 19, 20, 25, 28, 32, and 36 form group A. Claims 3, 12, and 21 form group B. Claims 4, 13, and 22 form group C. Claims 5, 14, 23, 29, 33, and 37 form group D. Claims 6, 15, and 24 form group E. Claims 8, 9, 17, 18, 26, 27, 30, 31, 34, 35, 38, and 39 form group F.

ARGUMENT

The Office Action rejects claims 1-7, 10-16, 19-25, 28, 29, 33, 36 & 37 under 35 U.S.C. § 103 as being unpatentable over *Smith et al.* (US Patent No. 6,477,703 B1) in view of *Chamberlain* (US Patent No. 6,427,227 B1). This rejection is respectfully traversed.

I. **The Prior Art Does Not Teach or Suggest Updating the Installer Program from Files in the Update and Installing the Update in the Program with the Updated Installer Program (Groups A-F)**

With respect to claim 1, the Office Action states:

Regarding claims, 1 Smith discloses, a method for updating code, the method (Col. 6:35-8:21), system (Fig. 1), comprising: providing an update to a plurality of versions of a program, wherein the program is updated by an installer program and a plurality of versions of the installer program exist (Smith, FIG. 1, step 110, FIG.3, step 330); determining whether a version of the installer program is incorrect with respect to the update (FIG.3, step 355,360); responsive to the version of the installer program being incorrect, updating the installer program from files in the update (FIG.6, step 610-670 also see associated text in Col: 5:40-6:18); and installing the update in the program with the updated installer program (FIG.5, 540). Smith doesn't explicitly disclose updating an installer program. However,

Chamberlain does disclose updating the installer program (FIG.3, items 31,37), also for product see (18:10-55). Therefore it would have been obvious to one of ordinary skills in the art at the time the invention was made to modify Smith with Chamberlain to attain the instant claimed invention because, "no software program is perfect and the release of numerous software patches to fix specific problems, and the release of new versions to fix or upgrade major problems are the norm" (Smith, Col. 1:20-23).

Office Action, dated December 30, 2003. Appellants respectfully disagree. *Smith* teaches a software patch selection tool providing a method and apparatus for identifying software patches for installation on a computer system in which current versions of software applications installed on a computer system are identified, as well as currently installed patches, products, and versions. An initial list of recommended patches is generated based on the combination of installed products and the validated list of datasheet selected products. See *Smith*, Abstract. The initial list of patches is checked for patches that are missing dependent patches and dependencies are added, as necessary. The initial list is also checked against itself and against a list of installed patches to remove redundant, obsolete, or outdated patches. Bad patches and patches that cause structural conflicts are also removed. A user may then inspect the list and deselect unwanted

patches. The user may also manually add other patches to the list. The tool then downloads the patches and orders them for execution. See *Smith*, col. 2, lines 17-41. *Smith* does not teach or suggest “updating the installer program from files in the update” and “installing the update in the program with the updated installer program,” as recited in claim 1.

Chamberlain teaches a mechanism for repairing an installed and patched application program if a patched resource needed by the application program becomes inadvertently deleted or otherwise unavailable to the application program. An installer receives a patch and performs the patching operations necessary to apply the patch to the installed product. *Chamberlain* states:

The patching operations may include adding new program files to the installed product, altering existing program files associated with components of the product, modifying entries within the system registry 39, or other patching operations.

Chamberlain, col. 11, lines 7-11. Also, the installer updates an installer registry to reflect the existence of the patch and to reflect the proper installed state of the product after the patch has been applied. *Chamberlain* states:

At step 414, the installer application 201 updates the installer directory 202 to reflect the proper installed state of the product after the patch has been applied.

Chamberlain, col. 10, lines 49-52.

The Office Action alleges that *Chamberlain* teaches updating the installer program in **Figure 3**, elements 31 and 37. This figure is shown below:

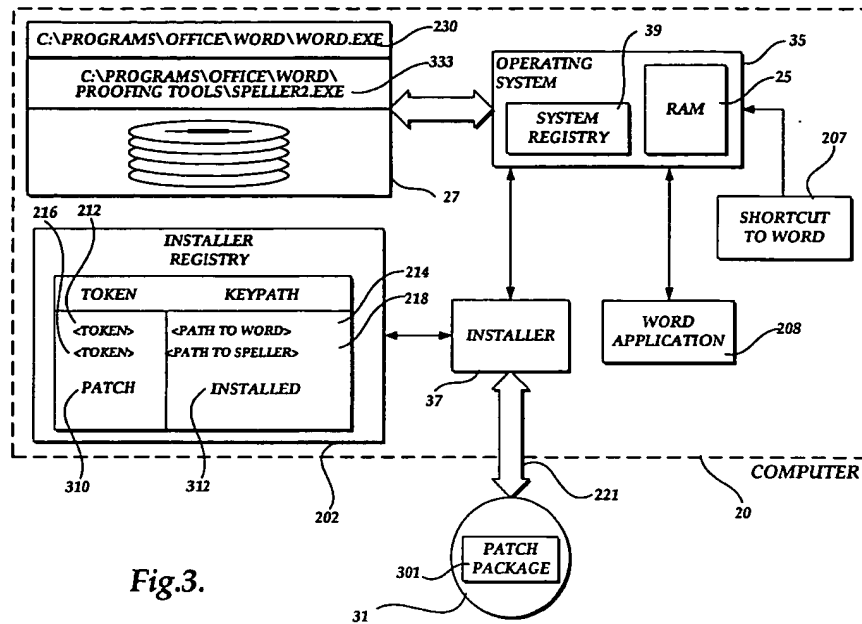


Fig.3.

Unfortunately, the reference number 37 does not appear anywhere in the text of *Chamberlain*. However, from the above referenced portions, it is clear that the patch package does not update the installer program as alleged in the Office Action. Rather, the installer program 201 updates software in hard drive 27 using the patch package 301.

In contradistinction, the present invention provides a method for updating code in a **software installer** program. Claim 1 recites:

1. A method for updating code, the method comprising:
 - providing an update to a plurality of versions of a program, wherein the program is updated by an installer program and a plurality of versions of the installer program exist;
 - determining whether a version of the installer program is incorrect with respect to the update;
 - responsive to the version of the installer program being incorrect, updating the installer program from files in the update; and
 - installing the update in the program with the updated installer program.

Neither *Smith* nor *Chamberlain* teaches or suggests “determining whether a version of the installer program is incorrect with respect to the update,” “responsive to the version of the installer program being incorrect, updating the installer program from files in the update,” and “installing the update in the program with the updated installer program,” as recited in claim 1. Since the applied references, taken alone or in combination, fail to teach or suggest each and every claim limitation, claim 1 cannot be rendered obvious by a combination of *Smith* and

Chamberlain. Therefore, the Office Action fails to establish a *prima facie* case of obviousness and the rejection of claim 1 should be withdrawn.

Independent claims 10, 19, 28, 32, and 36 recite subject matter addressed above with respect to claim 1 and are allowable for the same reasons. Since claims 2-7, 11-16, 20-25, 29, 33, and 37 depend from claims 1, 10, 19, 28, 32, and 36, the same distinctions between *Smith* and *Chamberlain* and the invention recited in claims 1, 10, 19, 28, 32, and 36 apply for these claims. Additionally, claims 2-7, 11-16, 20-25, 29, 33, and 37 recite other additional combinations of features not suggested by the references, alone or in combination.

II. The Prior Art Teaches Away From a Determination that the Version of the Installer is More Recent Than the Update Indicating that the Version of the Installer Program is Incorrect (Group B)

More particularly, with respect to claim 3, the Office action states:

Regarding claim 3 the method as recited in claim 1, wherein a determination that the version of the installer is more recent than the update indicates that the version of the installer program is incorrect (5:15-20 for recent see recommended superseded by installed patch, and 5:50-55, also see user and deselecting conflicting patches 5:48-51).

Office Action, dated December 30, 2003. Appellants respectfully disagree. The cited portion of *Smith* states:

All of the functionality of an older patch in a family tree is rolled into new patches in that family tree, thus, there is no reason to send and older patch, if a newer one is already on the system. At step 510, a decision is made as to whether there are any recommended patches superseded by installed patches. If so, the recommended patches are removed from recommended list (step 515).

Smith, col. 12-20. This portion of *Smith* actually teaches away from the invention recited in claim 3, because *Smith* teaches that a more recent version of a patch supersedes a less recent version. This is the opposite of the feature recited in claim 3, which recites “wherein a determination that the version of the installer is more recent than the update indicates that the version of the installer program is incorrect.” The cited portion of *Smith* also states:

If common modules exist between patches that are dependent on each other, they can also be ignored as they’re related although not in the same family tree (step 610). At step 620, any conflicts are

displayed to allow the user to choose between the conflicting patches. The user can “click” on the desired patch (resulting in a deselect of the other patch(es) from the final master list). The Final Master Patch List is then displayed (Step 630). Each patch is listed on a separate line with a checkbox in front to allow the user to deselect (therefore not send) a patch.

Smith, col. 5, lines 46-55. Thus, *Smith* teaches that patches that depend on one another (newer versions may depend upon older versions) are ignored and, thus, not found to be incorrect. Also, the cited portion of *Smith* teaches that a user can decide between conflicting patches. However, there is no teaching in *Smith* of a determining step, wherein a newer version of an update is found to be incorrect. *Chamberlain* also fails to make up for the deficiencies of *Smith*.

The applied references, taken alone or in combination, fail to teach or suggest each and every claim limitation; therefore, claim 3 cannot be rendered obvious by a combination of *Smith* and *Chamberlain*. Claims 12 and 21 recite subject matter addressed above with respect to claim 3 and are allowable for the same reasons.

III. The Prior Art Fails to Teach or Suggest that the Version of the Installer Program is Determined from a Single One of a Plurality of Files Contained within the Installer Program (Group C)

With respect to claim 4, the Office Action states:

Regarding claim 4 the method as recited in claim 1, wherein the version of the installer program is determined from a single one of a plurality of files contained within the installer program [Smith, FIG.1, step 110].

Office Action, dated December 30, 2003. Appellants respectfully disagree. The cited portion of *Smith* merely states, “IDENTIFY CURRENT VERSIONS OF INSTALLED SOFTWARE APPLICATIONS.” Neither the cited portion nor any other portion of *Smith* teaches or suggests determining the version of the **installer** program from a single one of the plurality of files contained within the installer program. The Examiner bears the burden of establishing a *prima facie* case of obviousness based on the prior art when rejecting claims under 35 U.S.C. § 103. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). The Office Action merely repeats the claim limitations and cites a seemingly arbitrary portion of one of the applied references without any analysis as to why the claim is allegedly obvious based on the cited portion. Thus, the Office Action fails to establish a *prima facie* case of obviousness. Appellants

submit that *Smith* and *Chamberlain*, taken alone or in combination, fail to teach or suggest determining the version of an installer program from one single file contained within the installer program. Claims 13 and 22 recite subject matter addressed above with respect to claim 4 and are allowable for the same reasons.

IV. **The Prior Art Fails to Teach or Suggest Overwriting Selected Files from the Installer Program with a Corresponding Updated File Extracted from the Update (Group D)**

With respect to claim 5, the Office Action states:

Regarding claim 5 the method as recited in claim 1, Smith discloses all the claimed limitations as applied in claim 1 above. Smith doesn't explicitly disclose wherein the updating step comprises: extracting installer files from the installer program into a directory and overwriting selected files from the installer program with a corresponding updated file extracted from the update. However, Chamberlain does disclose this in a similar configuration (Chamberlain 10, 50-55, also see Chamberlain, fig. 7, 705, 707, 709); and

Office Action, dated December 30, 2003. Appellants respectfully disagree. **Figure 7** of *Chamberlain* is as shown below:

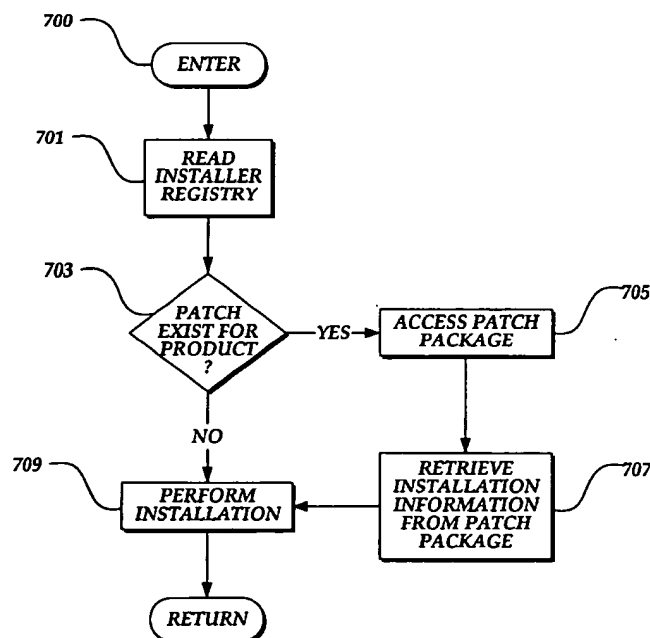


Fig.7.

Clearly, this figure does not teach or suggest “overwriting selected files **from the installer program** with a corresponding updated file extracted from the update,” as recited in claim 5.

With respect to performing installation, *Chamberlain* states:

At step 709, the installer application 201 performs an installation (or reinstallation) of the missing file to the location identified in the installer registry 202. In this example, the installation may include reading the original key file speller.exe 230 (**FIG. 2**) from the source 215, altering that program file with the patch bits from the patch package 301 to create the patched key file speller2.exe 333, and writing the patched key file speller2.exe 333 to the location identified by the keypath 218. It will be appreciated that reinstalling the missing program file may possibly include reinstalling other program files associated with the missing program file. In one example, the installer application 201 may install components of the product rather than individual program files, with each component having more than one associated program file.

Chamberlain, col. 17, lines 52-63. Clearly, this step performs installation, using the installer application, to an installed application, rather than to the installer application itself.

Furthermore, claim 5 recites, “packaging the updated files and remaining installer files into an updated installer program.” The Office Action does not address this feature due to an apparent typographical error. *Chamberlain* states:

At step 414, the installer application 201 updates the installer registry 202 to reflect the existence of the patch, and to reflect the proper installed state of the product after the patch has been applied. For example, the installer application 201 may add a patch entry 310 in the installer registry 202 indicating the existence of the patch.

Chamberlain, col. 10, lines 49-53. Appellants submit that *Smith* and *Chamberlain*, taken alone or in combination, fail to teach or suggest this feature. Therefore, a combination of *Smith* and *Chamberlain* would not result in the invention recited in claim 5. Claims 14, 23, 29, 33, and 37 recite subject matter addressed above with respect to claim 5 and are allowable for the same reasons. Also, claims 6, 15, and 24 depend from claims 5, 14, and 23, respectively, and are allowable by virtue of their dependency.

V. **The Prior Art Fails to Teach or Suggest Compressing the Updated Files and Remaining Installer Files to Produce an Updated Installer Program (Group E)**

More particularly, with respect to claim 6, the Office Action states:

Regarding claim 6 the method as recited in claim 5, wherein the packaging step comprises compressing the updated files and remaining installer files to produce an updated installer program (Chamberlain 1: 22-24).

Office Action, dated December 30, 2003. Appellant respectfully disagrees. The cited portion of *Chamberlain* states:

Often the program files are stored in a compressed format to conserve storage space.

Chamberlain, col. 1, lines 22-24. While compression of program files is generally known, the prior art as a whole fails to teach or suggest extracting (decompressing) files from an installer program, updating files from the installer program, and packaging and compressing the updated and remaining installer files to produce an updated installer program, as recited in claim 6. Even though *Chamberlain* teaches compression of program files in general, the combination of *Smith* and *Chamberlain* fails to arrive at the claimed invention. Therefore, claim 6 is not rendered obvious by the proposed combination of *Smith* and *Chamberlain*. Claims 15 and 24 recite subject matter addressed above with respect to claim 6 and are allowable for the same reasons.

Therefore, Appellants respectfully request that the rejection of claims 1-7, 10-16, 19-25, 28, 29, 33, 36 & 37 under 35 U.S.C. § 103 be overturned.

VI. **Forbes Does Not Make Up for the Deficiencies of Smith and Chamberlain (Group F)**

The Office Action rejects claims 8, 9, 17, 18, 26, 27, 30, 31, 34, 35, 38, and 39 under 35 U.S.C. § 103 as being unpatentable over *Smith* in view of *Chamberlain* and further in view of *Forbes et al.* (US Patent No. 6,381,742). This rejection is respectfully traversed.

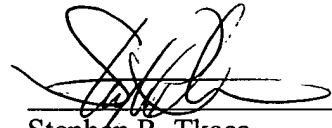
Claims 8, 9, 17, 18, 26, 27, 30, 31, 34, 35, 38, and 39 depend from claims 1, 10, 19, 28, 32, and 36 and are allowable by virtue of their dependency. While *Forbes* does teach that software packages are frequently written in common programming languages, such as Java, *Forbes* does not make up for the deficiencies of *Smith* and *Chamberlain*. More specifically, *Forbes*, like *Smith* and *Chamberlain*, fails to teach or suggest updating an installer program from files in an update, and installing an update in a program using the updated installer program.

Therefore, since the applied references, taken alone or in combination, fail to teach or suggest each and every claim limitation, claims 8, 9, 17, 18, 26, 27, 30, 31, 34, 35, 38, and 39 cannot be rendered obvious by the proposed combination of *Smith*, *Chamberlain*, and *Forbes*.

Therefore, Appellants respectfully request that the rejection of claims 8, 9, 17, 18, 26, 27, 30, 31, 34, 35, 38, and 39 under 35 U.S.C. § 103 be overturned.

VII. Conclusion

In view of the above, Appellants respectfully submit that the rejections of claims 1-39 are overcome. Accordingly, it is respectfully urged that the rejections of claims 1-39 not be sustained.



Stephen R. Tkacs
Reg. No. 46,430
Yee & Associates, P.C.
PO Box 802333
Dallas, TX 75380
(972) 367-2001

APPENDIX OF CLAIMS

The text of the claims involved in the appeal reads:

1. (Original): A method for updating code, the method comprising:

providing an update to a plurality of versions of a program, wherein the program is updated by an installer program and a plurality of versions of the installer program exist;

determining whether a version of the installer program is incorrect with respect to the update;

responsive to the version of the installer program being incorrect, updating the installer program from files in the update; and

installing the update in the program with the updated installer program.

2. (Original): The method as recited in claim 1, wherein a determination that the version of the installer is older than the update indicates that the version of the installer program is incorrect.

3. (Original): The method as recited in claim 1, wherein a determination that the version of the installer is more recent than the update indicates that the version of the installer program is incorrect.

4. (Original): The method as recited in claim 1, wherein the version of the installer program is determined from a single one of a plurality of files contained within the installer program.

5. (Original): The method as recited in claim 1, wherein the updating step comprises:

extracting installer files from the installer program into a directory;

overwriting selected files from the installer program with a corresponding updated file extracted from the update; and

packaging the updated files and remaining installer files into an updated installer program.

6. (Original): The method as recited in claim 5, wherein the packaging step comprises compressing the updated files and remaining installer files to produce an updated installer program.

7. (Original): The method as recited in claim 1, wherein the installer program comprises an install toolkit and the update comprises an update to the install toolkit.

8. (Original): The method as recited in claim 1, wherein the installer program and the update are written in an object-oriented programming language.

9. (Original): The method as recited in claim 1, wherein the installer program comprises a java install toolkit and the update comprises an update to the java install toolkit.

10. (Original): A computer program product in a computer readable media for use in a data processing system for updating code, the computer program product comprising:

first instructions for providing an update to a plurality of versions of a program, wherein the program is updated by an installer program and a plurality of versions of the installer program exist;

second instructions for determining whether a version of the installer program is incorrect with respect to the update;

third instructions, responsive to the version of the installer program being incorrect, for updating the installer program from files in the update; and

fourth instructions for installing the update in the program with the updated installer program.

11. (Original): The computer program product as recited in claim 10, wherein a determination that the version of the installer is older than the update indicates that the version of the installer program is incorrect.

12. (Original): The computer program product as recited in claim 10, wherein a determination that the version of the installer is more recent than the update indicates that the version of the installer program is incorrect.

13. (Original): The computer program product as recited in claim 10, wherein the version of the installer program is determined from a single one of a plurality of files contained within the installer program.

14. (Original): The computer program product as recited in claim 10, wherein the third instructions comprise:

fifth instructions for extracting installer files from the installer program into a directory;

sixth instructions for overwriting selected files from the installer program with a corresponding updated file extracted from the update; and

seventh instructions for packaging the updated files and remaining installer files into an updated installer program.

15. (Original): The computer program product as recited in claim 14, wherein the seventh instructions comprise compressing the updated files and remaining installer files to produce an updated installer program.

16. (Original): The computer program product as recited in claim 10, wherein the installer program comprises an install toolkit and the update comprises an update to the install toolkit.

17. (Original): The computer program product as recited in claim 10, wherein the installer program and the update are written in an object-oriented programming language.

18. (Original): The computer program product as recited in claim 10, wherein the installer program comprises a java install toolkit and the update comprises an update to the java install toolkit.

19. (Original): A system for updating code, the system comprising:

first means for providing an update to a plurality of versions of a program, wherein the program is updated by an installer program and a plurality of versions of the installer program exist;

second means for determining whether a version of the installer program is incorrect with respect to the update;

third means, responsive to the version of the installer program being incorrect, for updating the installer program from files in the update; and

fourth means for installing the update in the program with the updated installer program.

20. (Original): The system as recited in claim 19, wherein a determination that the version of the installer is older than the update indicates that the version of the installer program is incorrect.

21. (Original): The system as recited in claim 19, wherein a determination that the version of the installer is more recent than the update indicates that the version of the installer program is incorrect.

22. (Original): The system as recited in claim 19, wherein the version of the installer program is determined from a single one of a plurality of files contained within the installer program.

23. (Original): The system as recited in claim 19, wherein the third means comprise:

fifth means for extracting installer files from the installer program into a directory;

sixth means for overwriting selected files from the installer program with a corresponding updated file extracted from the update; and

seventh means for packaging the updated files and remaining installer files into an updated installer program.

24. (Original): The system as recited in claim 23, wherein the seventh means comprise compressing the updated files and remaining installer files to produce an updated installer program.

25. (Original): The system as recited in claim 19, wherein the installer program comprises an install toolkit and the update comprises an update to the install toolkit.

26. (Original): The system as recited in claim 19, wherein the installer program and the update are written in an object-oriented programming language.

27. (Original): The system as recited in claim 19, wherein the installer program comprises a java install toolkit and the update comprises an update to the java install toolkit.

28. (Original): A method in a data processing system for updating code, the method comprising:

receiving an update file for a program, wherein the update file includes an installer to update the program in which a plurality of versions of the installer exist;

determining whether the installer is a newer version of an existing installer for the program;

responsive to the installer being a new version of the existing installer, updating the existing installer to form an updated installer; and

installing code in the update file in the program using the updated installer.

29. (Original): The method as recited in claim 28, wherein the updating step comprises extracting files from the updated installer, overwriting corresponding older files extracted from the existing installer to form updated files, and repackaging the updated files to form the updated installer.

30. (Original): The method as recited in claim 28, wherein the update file, the program, the installer, the existing installer, and the update installer comprise object- oriented code.

31. (Original): The method as recited in claim 30, wherein the object-oriented code is java.

32. (Original): A computer program product in computer readable media for use in a data processing system for updating code, the computer program product comprising:

first instructions for receiving an update file for a program, wherein the update file includes an installer to update the program in which a plurality of versions of the installer exist;

second instructions for determining whether the installer is a newer version of an existing installer for the program;

third instructions, responsive to the installer being a new version of the existing installer, for updating the existing installer to form an updated installer; and

fourth instructions for installing code in the update file in the program using the updated installer.

33. (Original): The computer program product as recited in claim 32, wherein the third instructions comprise extracting files from the updated installer, overwriting corresponding older

files extracted from the existing installer to form updated files, and repackaging the updated files to form the updated installer.

34. (Original): The computer program product as recited in claim 32, wherein the update file, the program, the installer, the existing installer, and the update installer comprise object-oriented code.

35. (Original): The computer program product as recited in claim 34, wherein the object-oriented code is java.

36. (Original): A system for updating code, the system comprising:

first means for receiving an update file for a program, wherein the update file includes an installer to update the program in which a plurality of versions of the installer exist;

second means for determining whether the installer is a newer version of an existing installer for the program;

third means, responsive to the installer being a new version of the existing installer, for updating the existing installer to form an updated installer; and

fourth means for installing code in the update file in the program using the updated installer.

37. (Original): The system as recited in claim 36, wherein the third means comprise extracting files from the updated installer, overwriting corresponding older files extracted from the existing installer to form updated files, and repackaging the updated files to form the updated installer.

38. (Original): The system as recited in claim 36, wherein the update file, the program, the installer, the existing installer, and the update installer comprise object- oriented code.

39. (Original): The system as recited in claim 38, wherein the object-oriented code is java.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Curtis et al.** §
Serial No.: **09/578,750** §
Filed: **May 25, 2000** §
For: **Method of Applying an Update to** §
a Contained Collection of Program §
and Data Files Based Upon Versions §

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

ATTENTION: Board of Patent Appeals
and Interferences

Certificate of Mailing Under 37 C.F.R. § 1.8(a)

I hereby certify this correspondence is being deposited with the United States Postal Service as First Class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on May 28, 2004

By:

Amy Miller
Amy Miller

APPELLANT'S BRIEF (37 C.F.R. 1.192)

This brief is in furtherance of the Notice of Appeal, filed in this case on March 30, 2004.

The fees required under § 1.17(c), and any required petition for extension of time for filing this brief and fees therefore, are dealt with in the accompanying TRANSMITTAL OF APPEAL BRIEF.

This brief is transmitted in triplicate. (37 C.F.R. 1.192(a))

REAL PARTIES IN INTEREST

The real party in interest in this appeal is the following party: IBM Corporation

RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such appeals or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-39

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims canceled: NONE
2. Claims withdrawn from consideration but not canceled: NONE
3. Claims pending: 1-39
4. Claims allowed: NONE
5. Claims rejected: 1-39

C. CLAIMS ON APPEAL

The claims on appeal are: 1-39

STATUS OF AMENDMENTS

There are no amendments after final rejection.

SUMMARY OF INVENTION

The present invention provides a method, system, and apparatus for updating code of a software installer program. In a preferred embodiment a program, such as a patch, is provided to a plurality of versions of an install program, wherein the program is updated by an installation program and a plurality of versions of the installer program exist. See specification, page 12, lines 12-25; page 14, lines 7-18. Next, it is determined whether the version of the installer program is incorrect. See specification, page 14, lines 18-23. If the version is old, the installer program is updated from files in the patch. See specification, page 14, line 27, to page 15, line 4. The patch is then installed into the program using the updated installer program. See specification, page 15, lines 5-10.

ISSUES

The issues on appeal are as follows:

Whether claims 1-7, 10-16, 20-25, 28, 29, 33, 36, and 37 are unpatentable as being anticipated by *Smith et al.* (U.S. Patent No. 6,477,703) in view of Chamberlain (U.S. Patent No. 6,427,227).

Whether claims 8, 9, 17, 18, 26, 27, 30, 31, 34, 35, 38, and 39 are unpatentable as being obvious over *Smith et al.* in view of Chamberlain and further in view of *Forbes et al.* (U.S. Patent No. 6,381,742).

GROUPING OF CLAIMS

The claims on appeal do not stand or fall in a single group, but are grouped into the following groups for the reasons set forth in the Argument section below:

Claims 1, 2, 7, 10, 11, 16, 19, 20, 25, 28, 32, and 36 form group A. Claims 3, 12, and 21 form group B. Claims 4, 13, and 22 form group C. Claims 5, 14, 23, 29, 33, and 37 form group D. Claims 6, 15, and 24 form group E. Claims 8, 9, 17, 18, 26, 27, 30, 31, 34, 35, 38, and 39 form group F.

ARGUMENT

The Office Action rejects claims 1-7, 10-16, 19-25, 28, 29, 33, 36 & 37 under 35 U.S.C. § 103 as being unpatentable over *Smith et al.* (US Patent No. 6,477,703 B1) in view of *Chamberlain* (US Patent No. 6,427,227 B1). This rejection is respectfully traversed.

I. The Prior Art Does Not Teach or Suggest Updating the Installer Program from Files in the Update and Installing the Update in the Program with the Updated Installer Program (Groups A-F)

With respect to claim 1, the Office Action states:

Regarding claims, 1 Smith discloses, a method for updating code, the method (Col. 6:35-8:21), system (Fig. 1), comprising: providing an update to a plurality of versions of a program, wherein the program is updated by an installer program and a plurality of versions of the installer program exist (Smith, FIG. 1, step 110, FIG.3, step 330); determining whether a version of the installer program is incorrect with respect to the update (FIG.3, step 355,360); responsive to the version of the installer program being incorrect, updating the installer program from files in the update (FIG.6, step 610-670 also see associated text in Col: 5:40-6:18); and installing the update in the program with the updated installer program (FIG.5, 540). Smith doesn't explicitly disclose updating an installer program. However,

Chamberlain does disclose updating the installer program (FIG.3, items 31,37), also for product see (18:10-55). Therefore it would have been obvious to one of ordinary skills in the art at the time the invention was made to modify Smith with Chamberlain to attain the instant claimed invention because, "no software program is perfect and the release of numerous software patches to fix specific problems, and the release of new versions to fix or upgrade major problems are the norm" (Smith, Col. 1:20-23).

Office Action, dated December 30, 2003. Appellants respectfully disagree. *Smith* teaches a software patch selection tool providing a method and apparatus for identifying software patches for installation on a computer system in which current versions of software applications installed on a computer system are identified, as well as currently installed patches, products, and versions. An initial list of recommended patches is generated based on the combination of installed products and the validated list of datasheet selected products. See *Smith*, Abstract. The initial list of patches is checked for patches that are missing dependent patches and dependencies are added, as necessary. The initial list is also checked against itself and against a list of installed patches to remove redundant, obsolete, or outdated patches. Bad patches and patches that cause structural conflicts are also removed. A user may then inspect the list and deselect unwanted

patches. The user may also manually add other patches to the list. The tool then downloads the patches and orders them for execution. See *Smith*, col. 2, lines 17-41. *Smith* does not teach or suggest “updating the installer program from files in the update” and “installing the update in the program with the updated installer program,” as recited in claim 1.

Chamberlain teaches a mechanism for repairing an installed and patched application program if a patched resource needed by the application program becomes inadvertently deleted or otherwise unavailable to the application program. An installer receives a patch and performs the patching operations necessary to apply the patch to the installed product. *Chamberlain* states:

The patching operations may include adding new program files to the installed product, altering existing program files associated with components of the product, modifying entries within the system registry **39**, or other patching operations.

Chamberlain, col. 11, lines 7-11. Also, the installer updates an installer registry to reflect the existence of the patch and to reflect the proper installed state of the product after the patch has been applied. *Chamberlain* states:

At step **414**, the installer application **201** updates the installer directory **202** to reflect the proper installed state of the product after the patch has been applied.

Chamberlain, col. 10, lines 49-52.

The Office Action alleges that *Chamberlain* teaches updating the installer program in **Figure 3**, elements **31** and **37**. This figure is shown below:

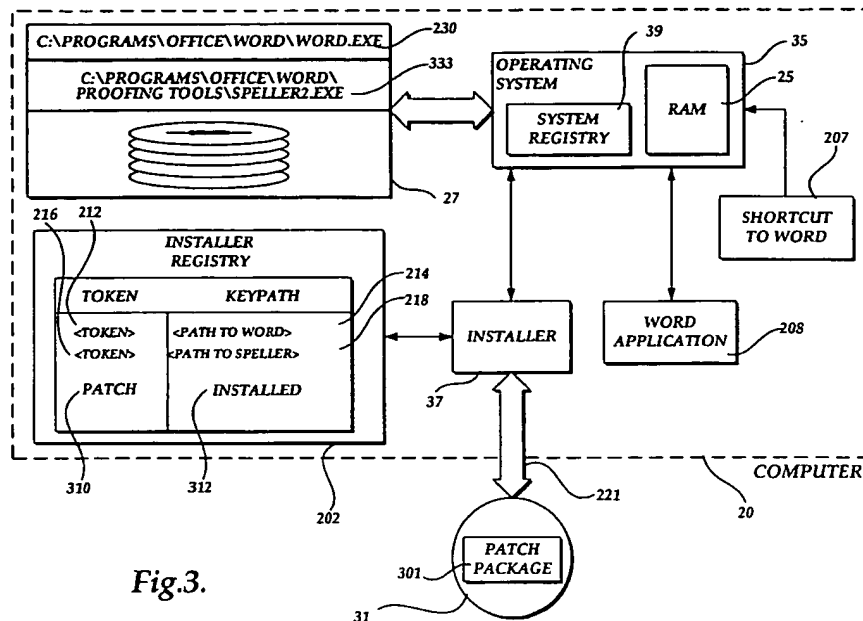


Fig.3.

Unfortunately, the reference number 37 does not appear anywhere in the text of *Chamberlain*. However, from the above referenced portions, it is clear that the patch package does not update the installer program as alleged in the Office Action. Rather, the installer program 201 updates software in hard drive 27 using the patch package 301.

In contradistinction, the present invention provides a method for updating code in a **software installer** program. Claim 1 recites:

1. A method for updating code, the method comprising:
 - providing an update to a plurality of versions of a program, wherein the program is updated by an installer program and a plurality of versions of the installer program exist;
 - determining whether a version of the installer program is incorrect with respect to the update;
 - responsive to the version of the installer program being incorrect, updating the installer program from files in the update; and
 - installing the update in the program with the updated installer program.

Neither *Smith* nor *Chamberlain* teaches or suggests “determining whether a version of the installer program is incorrect with respect to the update,” “responsive to the version of the installer program being incorrect, updating the installer program from files in the update,” and “installing the update in the program with the updated installer program,” as recited in claim 1. Since the applied references, taken alone or in combination, fail to teach or suggest each and every claim limitation, claim 1 cannot be rendered obvious by a combination of *Smith* and

Chamberlain. Therefore, the Office Action fails to establish a *prima facie* case of obviousness and the rejection of claim 1 should be withdrawn.

Independent claims 10, 19, 28, 32, and 36 recite subject matter addressed above with respect to claim 1 and are allowable for the same reasons. Since claims 2-7, 11-16, 20-25, 29, 33, and 37 depend from claims 1, 10, 19, 28, 32, and 36, the same distinctions between *Smith* and *Chamberlain* and the invention recited in claims 1, 10, 19, 28, 32, and 36 apply for these claims. Additionally, claims 2-7, 11-16, 20-25, 29, 33, and 37 recite other additional combinations of features not suggested by the references, alone or in combination.

II. The Prior Art Teaches Away From a Determination that the Version of the Installer is More Recent Than the Update Indicating that the Version of the Installer Program is Incorrect (Group B)

More particularly, with respect to claim 3, the Office action states:

Regarding claim 3 the method as recited in claim 1, wherein a determination that the version of the installer is more recent than the update indicates that the version of the installer program is incorrect (5:15-20 for recent see recommended superseded by installed patch, and 5:50-55, also see user and deselecting conflicting patches 5:48-51).

Office Action, dated December 30, 2003. Appellants respectfully disagree. The cited portion of *Smith* states:

All of the functionality of an older patch in a family tree is rolled into new patches in that family tree, thus, there is no reason to send and older patch, if a newer one is already on the system. At step 510, a decision is made as to whether there are any recommended patches superseded by installed patches. If so, the recommended patches are removed from recommended list (step 515).

Smith, col. 12-20. This portion of *Smith* actually teaches away from the invention recited in claim 3, because *Smith* teaches that a more recent version of a patch supersedes a less recent version. This is the opposite of the feature recited in claim 3, which recites “wherein a determination that the version of the installer is more recent than the update indicates that the version of the installer program is incorrect.” The cited portion of *Smith* also states:

If common modules exist between patches that are dependent on each other, they can also be ignored as they’re related although not in the same family tree (step 610). At step 620, any conflicts are

displayed to allow the user to choose between the conflicting patches. The user can “click” on the desired patch (resulting in a deselect of the other patch(es) from the final master list). The Final Master Patch List is then displayed (Step 630). Each patch is listed on a separate line with a checkbox in front to allow the user to deselect (therefore not send) a patch.

Smith, col. 5, lines 46-55. Thus, *Smith* teaches that patches that depend on one another (newer versions may depend upon older versions) are ignored and, thus, not found to be incorrect. Also, the cited portion of *Smith* teaches that a user can decide between conflicting patches. However, there is no teaching in *Smith* of a determining step, wherein a newer version of an update is found to be incorrect. *Chamberlain* also fails to make up for the deficiencies of *Smith*.

The applied references, taken alone or in combination, fail to teach or suggest each and every claim limitation; therefore, claim 3 cannot be rendered obvious by a combination of *Smith* and *Chamberlain*. Claims 12 and 21 recite subject matter addressed above with respect to claim 3 and are allowable for the same reasons.

III. The Prior Art Fails to Teach or Suggest that the Version of the Installer Program is Determined from a Single One of a Plurality of Files Contained within the Installer Program (Group C)

With respect to claim 4, the Office Action states:

Regarding claim 4 the method as recited in claim 1, wherein the version of the installer program is determined from a single one of a plurality of files contained within the installer program [Smith, FIG.1, step 110].

Office Action, dated December 30, 2003. Appellants respectfully disagree. The cited portion of *Smith* merely states, “IDENTIFY CURRENT VERSIONS OF INSTALLED SOFTWARE APPLICATIONS.” Neither the cited portion nor any other portion of *Smith* teaches or suggests determining the version of the **installer** program from a single one of the plurality of files contained within the installer program. The Examiner bears the burden of establishing a *prima facie* case of obviousness based on the prior art when rejecting claims under 35 U.S.C. § 103. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). The Office Action merely repeats the claim limitations and cites a seemingly arbitrary portion of one of the applied references without any analysis as to why the claim is allegedly obvious based on the cited portion. Thus, the Office Action fails to establish a *prima facie* case of obviousness. Appellants

submit that *Smith* and *Chamberlain*, taken alone or in combination, fail to teach or suggest determining the version of an installer program from one single file contained within the installer program. Claims 13 and 22 recite subject matter addressed above with respect to claim 4 and are allowable for the same reasons.

IV. **The Prior Art Fails to Teach or Suggest Overwriting Selected Files from the Installer Program with a Corresponding Updated File Extracted from the Update (Group D)**

With respect to claim 5, the Office Action states:

Regarding claim 5 the method as recited in claim 1, Smith discloses all the claimed limitations as applied in claim 1 above. Smith doesn't explicitly disclose wherein the updating step comprises: extracting installer files from the installer program into a directory and overwriting selected files from the installer program with a corresponding updated file extracted from the update. However, Chamberlain does disclose this in a similar configuration (Chamberlain 10, 50-55, also see Chamberlain, fig. 7, 705, 707, 709); and

Office Action, dated December 30, 2003. Appellants respectfully disagree. **Figure 7** of *Chamberlain* is as shown below:

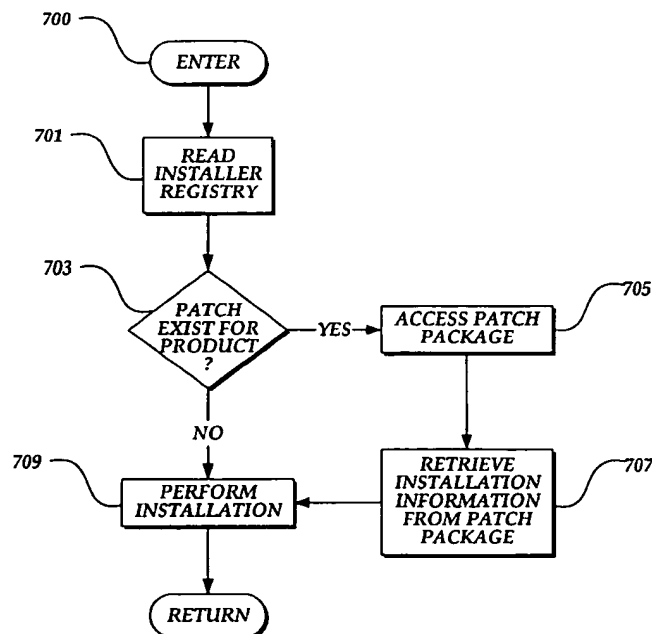


Fig.7.

Clearly, this figure does not teach or suggest “overwriting selected files **from the installer program** with a corresponding updated file extracted from the update,” as recited in claim 5.

With respect to performing installation, *Chamberlain* states:

At step 709, the installer application 201 performs an installation (or reinstallation) of the missing file to the location identified in the installer registry 202. In this example, the installation may include reading the original key file speller.exe 230 (FIG. 2) from the source 215, altering that program file with the patch bits from the patch package 301 to create the patched key file speller2.exe 333, and writing the patched key file speller2.exe 333 to the location identified by the keypath 218. It will be appreciated that reinstalling the missing program file may possibly include reinstalling other program files associated with the missing program file. In one example, the installer application 201 may install components of the product rather than individual program files, with each component having more than one associated program file.

Chamberlain, col. 17, lines 52-63. Clearly, this step performs installation, using the installer application, to an installed application, rather than to the installer application itself.

Furthermore, claim 5 recites, “packaging the updated files and remaining installer files into an updated installer program.” The Office Action does not address this feature due to an apparent typographical error. *Chamberlain* states:

At step 414, the installer application 201 updates the installer registry 202 to reflect the existence of the patch, and to reflect the proper installed state of the product after the patch has been applied. For example, the installer application 201 may add a patch entry 310 in the installer registry 202 indicating the existence of the patch.

Chamberlain, col. 10, lines 49-53. Appellants submit that *Smith* and *Chamberlain*, taken alone or in combination, fail to teach or suggest this feature. Therefore, a combination of *Smith* and *Chamberlain* would not result in the invention recited in claim 5. Claims 14, 23, 29, 33, and 37 recite subject matter addressed above with respect to claim 5 and are allowable for the same reasons. Also, claims 6, 15, and 24 depend from claims 5, 14, and 23, respectively, and are allowable by virtue of their dependency.

V. The Prior Art Fails to Teach or Suggest Compressing the Updated Files and Remaining Installer Files to Produce an Updated Installer Program (Group E)

More particularly, with respect to claim 6, the Office Action states:

Regarding claim 6 the method as recited in claim 5, wherein the packaging step comprises compressing the updated files and remaining installer files to produce an updated installer program (Chamberlain 1: 22-24).

Office Action, dated December 30, 2003. Appellant respectfully disagrees. The cited portion of *Chamberlain* states:

Often the program files are stored in a compressed format to conserve storage space.

Chamberlain, col. 1, lines 22-24. While compression of program files is generally known, the prior art as a whole fails to teach or suggest extracting (decompressing) files from an installer program, updating files from the installer program, and packaging and compressing the updated and remaining installer files to produce an updated installer program, as recited in claim 6. Even though *Chamberlain* teaches compression of program files in general, the combination of *Smith* and *Chamberlain* fails to arrive at the claimed invention. Therefore, claim 6 is not rendered obvious by the proposed combination of *Smith* and *Chamberlain*. Claims 15 and 24 recite subject matter addressed above with respect to claim 6 and are allowable for the same reasons.

Therefore, Appellants respectfully request that the rejection of claims 1-7, 10-16, 19-25, 28, 29, 33, 36 & 37 under 35 U.S.C. § 103 be overturned.

VI. Forbes Does Not Make Up for the Deficiencies of Smith and Chamberlain (Group F)

The Office Action rejects claims 8, 9, 17, 18, 26, 27, 30, 31, 34, 35, 38, and 39 under 35 U.S.C. § 103 as being unpatentable over *Smith* in view of *Chamberlain* and further in view of *Forbes et al.* (US Patent No. 6,381,742). This rejection is respectfully traversed.

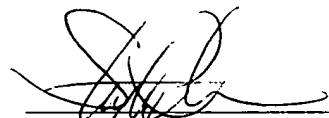
Claims 8, 9, 17, 18, 26, 27, 30, 31, 34, 35, 38, and 39 depend from claims 1, 10, 19, 28, 32, and 36 and are allowable by virtue of their dependency. While *Forbes* does teach that software packages are frequently written in common programming languages, such as Java, *Forbes* does not make up for the deficiencies of *Smith* and *Chamberlain*. More specifically, *Forbes*, like *Smith* and *Chamberlain*, fails to teach or suggest updating an installer program from files in an update, and installing an update in a program using the updated installer program.

Therefore, since the applied references, taken alone or in combination, fail to teach or suggest each and every claim limitation, claims 8, 9, 17, 18, 26, 27, 30, 31, 34, 35, 38, and 39 cannot be rendered obvious by the proposed combination of *Smith*, *Chamberlain*, and *Forbes*.

Therefore, Appellants respectfully request that the rejection of claims 8, 9, 17, 18, 26, 27, 30, 31, 34, 35, 38, and 39 under 35 U.S.C. § 103 be overturned.

VII. Conclusion

In view of the above, Appellants respectfully submit that the rejections of claims 1-39 are overcome. Accordingly, it is respectfully urged that the rejections of claims 1-39 not be sustained.



Stephen R. Tkacs
Reg. No. 46,430
Yee & Associates, P.C.
PO Box 802333
Dallas, TX 75380
(972) 367-2001

APPENDIX OF CLAIMS

The text of the claims involved in the appeal reads:

1. (Original): A method for updating code, the method comprising:

providing an update to a plurality of versions of a program, wherein the program is updated by an installer program and a plurality of versions of the installer program exist;

determining whether a version of the installer program is incorrect with respect to the update;

responsive to the version of the installer program being incorrect, updating the installer program from files in the update; and

installing the update in the program with the updated installer program.

2. (Original): The method as recited in claim 1, wherein a determination that the version of the installer is older than the update indicates that the version of the installer program is incorrect.

3. (Original): The method as recited in claim 1, wherein a determination that the version of the installer is more recent than the update indicates that the version of the installer program is incorrect.

4. (Original): The method as recited in claim 1, wherein the version of the installer program is determined from a single one of a plurality of files contained within the installer program.

5. (Original): The method as recited in claim 1, wherein the updating step comprises:

extracting installer files from the installer program into a directory;

overwriting selected files from the installer program with a corresponding updated file extracted from the update; and

packaging the updated files and remaining installer files into an updated installer program.

6. (Original): The method as recited in claim 5, wherein the packaging step comprises compressing the updated files and remaining installer files to produce an updated installer program.

7. (Original): The method as recited in claim 1, wherein the installer program comprises an install toolkit and the update comprises an update to the install toolkit.

8. (Original): The method as recited in claim 1, wherein the installer program and the update are written in an object-oriented programming language.

9. (Original): The method as recited in claim 1, wherein the installer program comprises a java install toolkit and the update comprises an update to the java install toolkit.

10. (Original): A computer program product in a computer readable media for use in a data processing system for updating code, the computer program product comprising:

first instructions for providing an update to a plurality of versions of a program, wherein the program is updated by an installer program and a plurality of versions of the installer program exist;

second instructions for determining whether a version of the installer program is incorrect with respect to the update;

third instructions, responsive to the version of the installer program being incorrect, for updating the installer program from files in the update; and

fourth instructions for installing the update in the program with the updated installer program.

11. (Original): The computer program product as recited in claim 10, wherein a determination that the version of the installer is older than the update indicates that the version of the installer program is incorrect.

12. (Original): The computer program product as recited in claim 10, wherein a determination that the version of the installer is more recent than the update indicates that the version of the installer program is incorrect.

13. (Original): The computer program product as recited in claim 10, wherein the version of the installer program is determined from a single one of a plurality of files contained within the installer program.

14. (Original): The computer program product as recited in claim 10, wherein the third instructions comprise:

fifth instructions for extracting installer files from the installer program into a directory;

sixth instructions for overwriting selected files from the installer program with a corresponding updated file extracted from the update; and

seventh instructions for packaging the updated files and remaining installer files into an updated installer program.

15. (Original): The computer program product as recited in claim 14, wherein the seventh instructions comprise compressing the updated files and remaining installer files to produce an updated installer program.

16. (Original): The computer program product as recited in claim 10, wherein the installer program comprises an install toolkit and the update comprises an update to the install toolkit.

17. (Original): The computer program product as recited in claim 10, wherein the installer program and the update are written in an object-oriented programming language.

18. (Original): The computer program product as recited in claim 10, wherein the installer program comprises a java install toolkit and the update comprises an update to the java install toolkit.

19. (Original): A system for updating code, the system comprising:

first means for providing an update to a plurality of versions of a program, wherein the program is updated by an installer program and a plurality of versions of the installer program exist;

second means for determining whether a version of the installer program is incorrect with respect to the update;

third means, responsive to the version of the installer program being incorrect, for updating the installer program from files in the update; and

fourth means for installing the update in the program with the updated installer program.

20. (Original): The system as recited in claim 19, wherein a determination that the version of the installer is older than the update indicates that the version of the installer program is incorrect.

21. (Original): The system as recited in claim 19, wherein a determination that the version of the installer is more recent than the update indicates that the version of the installer program is incorrect.

22. (Original): The system as recited in claim 19, wherein the version of the installer program is determined from a single one of a plurality of files contained within the installer program.

23. (Original): The system as recited in claim 19, wherein the third means comprise:

fifth means for extracting installer files from the installer program into a directory;

sixth means for overwriting selected files from the installer program with a corresponding updated file extracted from the update; and

seventh means for packaging the updated files and remaining installer files into an updated installer program.

24. (Original): The system as recited in claim 23, wherein the seventh means comprise compressing the updated files and remaining installer files to produce an updated installer program.

25. (Original): The system as recited in claim 19, wherein the installer program comprises an install toolkit and the update comprises an update to the install toolkit.

26. (Original): The system as recited in claim 19, wherein the installer program and the update are written in an object-oriented programming language.

27. (Original): The system as recited in claim 19, wherein the installer program comprises a java install toolkit and the update comprises an update to the java install toolkit.

28. (Original): A method in a data processing system for updating code, the method comprising:

receiving an update file for a program, wherein the update file includes an installer to update the program in which a plurality of versions of the installer exist;

determining whether the installer is a newer version of an existing installer for the program;

responsive to the installer being a new version of the existing installer, updating the existing installer to form an updated installer; and

installing code in the update file in the program using the updated installer.

29. (Original): The method as recited in claim 28, wherein the updating step comprises extracting files from the updated installer, overwriting corresponding older files extracted from the existing installer to form updated files, and repackaging the updated files to form the updated installer.

30. (Original): The method as recited in claim 28, wherein the update file, the program, the installer, the existing installer, and the update installer comprise object- oriented code.

31. (Original): The method as recited in claim 30, wherein the object-oriented code is java.

32. (Original): A computer program product in computer readable media for use in a data processing system for updating code, the computer program product comprising:

first instructions for receiving an update file for a program, wherein the update file includes an installer to update the program in which a plurality of versions of the installer exist;

second instructions for determining whether the installer is a newer version of an existing installer for the program;

third instructions, responsive to the installer being a new version of the existing installer, for updating the existing installer to form an updated installer; and

fourth instructions for installing code in the update file in the program using the updated installer.

33. (Original): The computer program product as recited in claim 32, wherein the third instructions comprise extracting files from the updated installer, overwriting corresponding older

files extracted from the existing installer to form updated files, and repackaging the updated files to form the updated installer.

34. (Original): The computer program product as recited in claim 32, wherein the update file, the program, the installer, the existing installer, and the update installer comprise object-oriented code.

35. (Original): The computer program product as recited in claim 34, wherein the object-oriented code is java.

36. (Original): A system for updating code, the system comprising:

first means for receiving an update file for a program, wherein the update file includes an installer to update the program in which a plurality of versions of the installer exist;

second means for determining whether the installer is a newer version of an existing installer for the program;

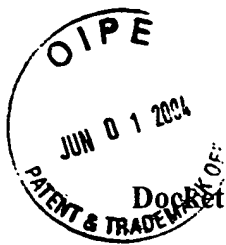
third means, responsive to the installer being a new version of the existing installer, for updating the existing installer to form an updated installer; and

fourth means for installing code in the update file in the program using the updated installer.

37. (Original): The system as recited in claim 36, wherein the third means comprise extracting files from the updated installer, overwriting corresponding older files extracted from the existing installer to form updated files, and repackaging the updated files to form the updated installer.

38. (Original): The system as recited in claim 36, wherein the update file, the program, the installer, the existing installer, and the update installer comprise object- oriented code.

39. (Original): The system as recited in claim 38, wherein the object-oriented code is java.



Doc. No. AUS9-2000-0214-US1

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Curtis et al.**

Serial No.: **09/578,750**

Filed: **May 25, 2000**

For: **Method of Applying an Update to
a Contained Collection of Program
and Data Files Based Upon Versions**

§
§
§
§
§
§
§

Group Art Unit: **2122**

Examiner: **Kendall, Chuck O.**

**Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450**

**ATTENTION: Board of Patent Appeals
and Interferences**

Certificate of Mailing Under 37 C.F.R. § 1.8(a)

I hereby certify this correspondence is being deposited with the United States Postal Service as First Class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on May 26, 2004

By:

Amy Miller
Amy Miller

APPELLANT'S BRIEF (37 C.F.R. 1.192)

This brief is in furtherance of the Notice of Appeal, filed in this case on March 30, 2004.

The fees required under § 1.17(c), and any required petition for extension of time for filing this brief and fees therefore, are dealt with in the accompanying TRANSMITTAL OF APPEAL BRIEF.

This brief is transmitted in triplicate. (37 C.F.R. 1.192(a))

REAL PARTIES IN INTEREST

The real party in interest in this appeal is the following party: IBM Corporation

RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such appeals or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-39

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims canceled: NONE
2. Claims withdrawn from consideration but not canceled: NONE
3. Claims pending: 1-39
4. Claims allowed: NONE
5. Claims rejected: 1-39

C. CLAIMS ON APPEAL

The claims on appeal are: 1-39

STATUS OF AMENDMENTS

There are no amendments after final rejection.

SUMMARY OF INVENTION

The present invention provides a method, system, and apparatus for updating code of a software installer program. In a preferred embodiment a program, such as a patch, is provided to a plurality of versions of an install program, wherein the program is updated by an installation program and a plurality of versions of the installer program exist. See specification, page 12, lines 12-25; page 14, lines 7-18. Next, it is determined whether the version of the installer program is incorrect. See specification, page 14, lines 18-23. If the version is old, the installer program is updated from files in the patch. See specification, page 14, line 27, to page 15, line 4. The patch is then installed into the program using the updated installer program. See specification, page 15, lines 5-10.

ISSUES

The issues on appeal are as follows:

Whether claims 1-7, 10-16, 20-25, 28, 29, 33, 36, and 37 are unpatentable as being anticipated by *Smith et al.* (U.S. Patent No. 6,477,703) in view of Chamberlain (U.S. Patent No. 6,427,227).

Whether claims 8, 9, 17, 18, 26, 27, 30, 31, 34, 35, 38, and 39 are unpatentable as being obvious over *Smith et al.* in view of Chamberlain and further in view of *Forbes et al.* (U.S. Patent No. 6,381,742).

GROUPING OF CLAIMS

The claims on appeal do not stand or fall in a single group, but are grouped into the following groups for the reasons set forth in the Argument section below:

Claims 1, 2, 7, 10, 11, 16, 19, 20, 25, 28, 32, and 36 form group A. Claims 3, 12, and 21 form group B. Claims 4, 13, and 22 form group C. Claims 5, 14, 23, 29, 33, and 37 form group D. Claims 6, 15, and 24 form group E. Claims 8, 9, 17, 18, 26, 27, 30, 31, 34, 35, 38, and 39 form group F.

ARGUMENT

The Office Action rejects claims 1-7, 10-16, 19-25, 28, 29, 33, 36 & 37 under 35 U.S.C. § 103 as being unpatentable over *Smith et al.* (US Patent No. 6,477,703 B1) in view of *Chamberlain* (US Patent No. 6,427,227 B1). This rejection is respectfully traversed.

I. The Prior Art Does Not Teach or Suggest Updating the Installer Program from Files in the Update and Installing the Update in the Program with the Updated Installer Program (Groups A-F)

With respect to claim 1, the Office Action states:

Regarding claims, 1 Smith discloses, a method for updating code, the method (Col. 6:35-8:21), system (Fig. 1), comprising: providing an update to a plurality of versions of a program, wherein the program is updated by an installer program and a plurality of versions of the installer program exist (Smith, FIG. 1, step 110, FIG.3, step 330); determining whether a version of the installer program is incorrect with respect to the update (FIG.3, step 355,360); responsive to the version of the installer program being incorrect, updating the installer program from files in the update (FIG.6, step 610-670 also see associated text in Col: 5:40-6:18); and installing the update in the program with the updated installer program (FIG.5, 540). Smith doesn't explicitly disclose updating an installer program. However,

Chamberlain does disclose updating the installer program (FIG.3, items 31,37), also for product see (18:10-55). Therefore it would have been obvious to one of ordinary skills in the art at the time the invention was made to modify Smith with Chamberlain to attain the instant claimed invention because, "no software program is perfect and the release of numerous software patches to fix specific problems, and the release of new versions to fix or upgrade major problems are the norm" (Smith, Col. 1:20-23).

Office Action, dated December 30, 2003. Appellants respectfully disagree. *Smith* teaches a software patch selection tool providing a method and apparatus for identifying software patches for installation on a computer system in which current versions of software applications installed on a computer system are identified, as well as currently installed patches, products, and versions. An initial list of recommended patches is generated based on the combination of installed products and the validated list of datasheet selected products. See *Smith*, Abstract. The initial list of patches is checked for patches that are missing dependent patches and dependencies are added, as necessary. The initial list is also checked against itself and against a list of installed patches to remove redundant, obsolete, or outdated patches. Bad patches and patches that cause structural conflicts are also removed. A user may then inspect the list and deselect unwanted

patches. The user may also manually add other patches to the list. The tool then downloads the patches and orders them for execution. See *Smith*, col. 2, lines 17-41. *Smith* does not teach or suggest “updating the installer program from files in the update” and “installing the update in the program with the updated installer program,” as recited in claim 1.

Chamberlain teaches a mechanism for repairing an installed and patched application program if a patched resource needed by the application program becomes inadvertently deleted or otherwise unavailable to the application program. An installer receives a patch and performs the patching operations necessary to apply the patch to the installed product. *Chamberlain* states:

The patching operations may include adding new program files to the installed product, altering existing program files associated with components of the product, modifying entries within the system registry **39**, or other patching operations.

Chamberlain, col. 11, lines 7-11. Also, the installer updates an installer registry to reflect the existence of the patch and to reflect the proper installed state of the product after the patch has been applied. *Chamberlain* states:

At step **414**, the installer application **201** updates the installer directory **202** to reflect the proper installed state of the product after the patch has been applied.

Chamberlain, col. 10, lines 49-52.

The Office Action alleges that *Chamberlain* teaches updating the installer program in **Figure 3**, elements **31** and **37**. This figure is shown below:

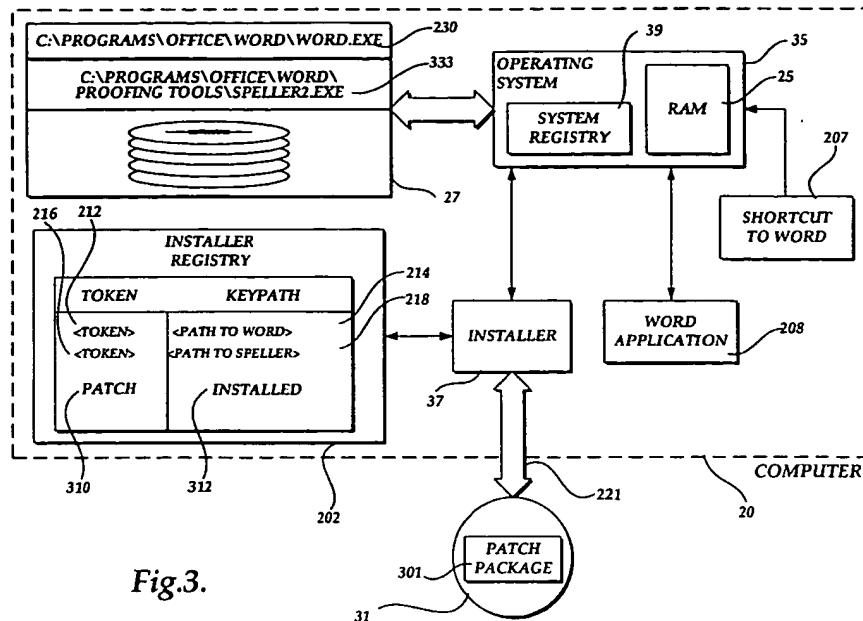


Fig.3.

Unfortunately, the reference number 37 does not appear anywhere in the text of *Chamberlain*. However, from the above referenced portions, it is clear that the patch package does not update the installer program as alleged in the Office Action. Rather, the installer program 201 updates software in hard drive 27 using the patch package 301.

In contradistinction, the present invention provides a method for updating code in a **software installer** program. Claim 1 recites:

1. A method for updating code, the method comprising:
 - providing an update to a plurality of versions of a program, wherein the program is updated by an installer program and a plurality of versions of the installer program exist;
 - determining whether a version of the installer program is incorrect with respect to the update;
 - responsive to the version of the installer program being incorrect, updating the installer program from files in the update; and
 - installing the update in the program with the updated installer program.

Neither *Smith* nor *Chamberlain* teaches or suggests “determining whether a version of the installer program is incorrect with respect to the update,” “responsive to the version of the installer program being incorrect, updating the installer program from files in the update,” and “installing the update in the program with the updated installer program,” as recited in claim 1. Since the applied references, taken alone or in combination, fail to teach or suggest each and every claim limitation, claim 1 cannot be rendered obvious by a combination of *Smith* and

Chamberlain. Therefore, the Office Action fails to establish a *prima facie* case of obviousness and the rejection of claim 1 should be withdrawn.

Independent claims 10, 19, 28, 32, and 36 recite subject matter addressed above with respect to claim 1 and are allowable for the same reasons. Since claims 2-7, 11-16, 20-25, 29, 33, and 37 depend from claims 1, 10, 19, 28, 32, and 36, the same distinctions between *Smith* and *Chamberlain* and the invention recited in claims 1, 10, 19, 28, 32, and 36 apply for these claims. Additionally, claims 2-7, 11-16, 20-25, 29, 33, and 37 recite other additional combinations of features not suggested by the references, alone or in combination.

II. The Prior Art Teaches Away From a Determination that the Version of the Installer is More Recent Than the Update Indicating that the Version of the Installer Program is Incorrect (Group B)

More particularly, with respect to claim 3, the Office action states:

Regarding claim 3 the method as recited in claim 1, wherein a determination that the version of the installer is more recent than the update indicates that the version of the installer program is incorrect (5:15-20 for recent see recommended superseded by installed patch, and 5:50-55, also see user and deselecting conflicting patches 5:48-51).

Office Action, dated December 30, 2003. Appellants respectfully disagree. The cited portion of *Smith* states:

All of the functionality of an older patch in a family tree is rolled into new patches in that family tree, thus, there is no reason to send and older patch, if a newer one is already on the system. At step **510**, a decision is made as to whether there are any recommended patches superseded by installed patches. If so, the recommended patches are removed from recommended list (step **515**).

Smith, col. 12-20. This portion of *Smith* actually teaches away from the invention recited in claim 3, because *Smith* teaches that a more recent version of a patch supersedes a less recent version. This is the opposite of the feature recited in claim 3, which recites “wherein a determination that the version of the installer is more recent than the update indicates that the version of the installer program is incorrect.” The cited portion of *Smith* also states:

If common modules exist between patches that are dependent on each other, they can also be ignored as they’re related although not in the same family tree (step **610**). At step **620**, any conflicts are

displayed to allow the user to choose between the conflicting patches. The user can “click” on the desired patch (resulting in a deselect of the other patch(es) from the final master list). The Final Master Patch List is then displayed (Step 630). Each patch is listed on a separate line with a checkbox in front to allow the user to deselect (therefore not send) a patch.

Smith, col. 5, lines 46-55. Thus, *Smith* teaches that patches that depend on one another (newer versions may depend upon older versions) are ignored and, thus, not found to be incorrect. Also, the cited portion of *Smith* teaches that a user can decide between conflicting patches. However, there is no teaching in *Smith* of a determining step, wherein a newer version of an update is found to be incorrect. *Chamberlain* also fails to make up for the deficiencies of *Smith*.

The applied references, taken alone or in combination, fail to teach or suggest each and every claim limitation; therefore, claim 3 cannot be rendered obvious by a combination of *Smith* and *Chamberlain*. Claims 12 and 21 recite subject matter addressed above with respect to claim 3 and are allowable for the same reasons.

III. The Prior Art Fails to Teach or Suggest that the Version of the Installer Program is Determined from a Single One of a Plurality of Files Contained within the Installer Program (Group C)

With respect to claim 4, the Office Action states:

Regarding claim 4 the method as recited in claim 1, wherein the version of the installer program is determined from a single one of a plurality of files contained within the installer program [Smith, FIG.1, step 110].

Office Action, dated December 30, 2003. Appellants respectfully disagree. The cited portion of *Smith* merely states, “IDENTIFY CURRENT VERSIONS OF INSTALLED SOFTWARE APPLICATIONS.” Neither the cited portion nor any other portion of *Smith* teaches or suggests determining the version of the **installer** program from a single one of the plurality of files contained within the installer program. The Examiner bears the burden of establishing a *prima facie* case of obviousness based on the prior art when rejecting claims under 35 U.S.C. § 103. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). The Office Action merely repeats the claim limitations and cites a seemingly arbitrary portion of one of the applied references without any analysis as to why the claim is allegedly obvious based on the cited portion. Thus, the Office Action fails to establish a *prima facie* case of obviousness. Appellants

submit that *Smith* and *Chamberlain*, taken alone or in combination, fail to teach or suggest determining the version of an installer program from one single file contained within the installer program. Claims 13 and 22 recite subject matter addressed above with respect to claim 4 and are allowable for the same reasons.

IV. The Prior Art Fails to Teach or Suggest Overwriting Selected Files from the Installer Program with a Corresponding Updated File Extracted from the Update (Group D)

With respect to claim 5, the Office Action states:

Regarding claim 5 the method as recited in claim 1, Smith discloses all the claimed limitations as applied in claim 1 above. Smith doesn't explicitly disclose wherein the updating step comprises: extracting installer files from the installer program into a directory and overwriting selected files from the installer program with a corresponding updated file extracted from the update. However, Chamberlain does disclose this in a similar configuration (Chamberlain 10, 50-55, also see Chamberlain, fig. 7, 705, 707, 709); and

Office Action, dated December 30, 2003. Appellants respectfully disagree. **Figure 7** of *Chamberlain* is as shown below:

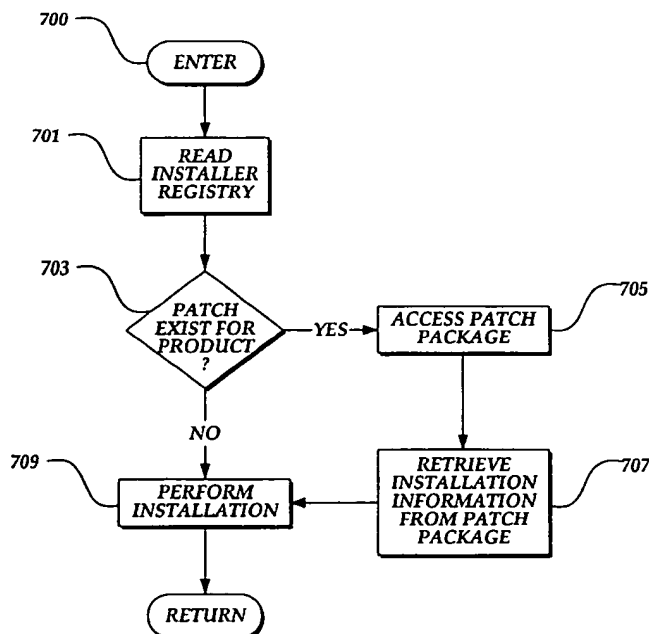


Fig.7.

Clearly, this figure does not teach or suggest “overwriting selected files **from the installer program** with a corresponding updated file extracted from the update,” as recited in claim 5.

With respect to performing installation, *Chamberlain* states:

At step 709, the installer application 201 performs an installation (or reinstallation) of the missing file to the location identified in the installer registry 202. In this example, the installation may include reading the original key file speller.exe 230 (FIG. 2) from the source 215, altering that program file with the patch bits from the patch package 301 to create the patched key file speller2.exe 333, and writing the patched key file speller2.exe 333 to the location identified by the keypath 218. It will be appreciated that reinstalling the missing program file may possibly include reinstalling other program files associated with the missing program file. In one example, the installer application 201 may install components of the product rather than individual program files, with each component having more than one associated program file.

Chamberlain, col. 17, lines 52-63. Clearly, this step performs installation, using the installer application, to an installed application, rather than to the installer application itself.

Furthermore, claim 5 recites, “packaging the updated files and remaining installer files into an updated installer program.” The Office Action does not address this feature due to an apparent typographical error. *Chamberlain* states:

At step 414, the installer application 201 updates the installer registry 202 to reflect the existence of the patch, and to reflect the proper installed state of the product after the patch has been applied. For example, the installer application 201 may add a patch entry 310 in the installer registry 202 indicating the existence of the patch.

Chamberlain, col. 10, lines 49-53. Appellants submit that *Smith* and *Chamberlain*, taken alone or in combination, fail to teach or suggest this feature. Therefore, a combination of *Smith* and *Chamberlain* would not result in the invention recited in claim 5. Claims 14, 23, 29, 33, and 37 recite subject matter addressed above with respect to claim 5 and are allowable for the same reasons. Also, claims 6, 15, and 24 depend from claims 5, 14, and 23, respectively, and are allowable by virtue of their dependency.

V. **The Prior Art Fails to Teach or Suggest Compressing the Updated Files and Remaining Installer Files to Produce an Updated Installer Program (Group E)**

More particularly, with respect to claim 6, the Office Action states:

Regarding claim 6 the method as recited in claim 5, wherein the packaging step comprises compressing the updated files and remaining installer files to produce an updated installer program (Chamberlain 1: 22-24).

Office Action, dated December 30, 2003. Appellant respectfully disagrees. The cited portion of *Chamberlain* states:

Often the program files are stored in a compressed format to conserve storage space.

Chamberlain, col. 1, lines 22-24. While compression of program files is generally known, the prior art as a whole fails to teach or suggest extracting (decompressing) files from an installer program, updating files from the installer program, and packaging and compressing the updated and remaining installer files to produce an updated installer program, as recited in claim 6. Even though *Chamberlain* teaches compression of program files in general, the combination of *Smith* and *Chamberlain* fails to arrive at the claimed invention. Therefore, claim 6 is not rendered obvious by the proposed combination of *Smith* and *Chamberlain*. Claims 15 and 24 recite subject matter addressed above with respect to claim 6 and are allowable for the same reasons.

Therefore, Appellants respectfully request that the rejection of claims 1-7, 10-16, 19-25, 28, 29, 33, 36 & 37 under 35 U.S.C. § 103 be overturned.

VI. **Forbes Does Not Make Up for the Deficiencies of Smith and Chamberlain (Group E)**

The Office Action rejects claims 8, 9, 17, 18, 26, 27, 30, 31, 34, 35, 38, and 39 under 35 U.S.C. § 103 as being unpatentable over *Smith* in view of *Chamberlain* and further in view of *Forbes et al.* (US Patent No. 6,381,742). This rejection is respectfully traversed.

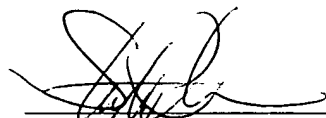
Claims 8, 9, 17, 18, 26, 27, 30, 31, 34, 35, 38, and 39 depend from claims 1, 10, 19, 28, 32, and 36 and are allowable by virtue of their dependency. While *Forbes* does teach that software packages are frequently written in common programming languages, such as Java, *Forbes* does not make up for the deficiencies of *Smith* and *Chamberlain*. More specifically, *Forbes*, like *Smith* and *Chamberlain*, fails to teach or suggest updating an installer program from files in an update, and installing an update in a program using the updated installer program.

Therefore, since the applied references, taken alone or in combination, fail to teach or suggest each and every claim limitation, claims 8, 9, 17, 18, 26, 27, 30, 31, 34, 35, 38, and 39 cannot be rendered obvious by the proposed combination of *Smith*, *Chamberlain*, and *Forbes*.

Therefore, Appellants respectfully request that the rejection of claims 8, 9, 17, 18, 26, 27, 30, 31, 34, 35, 38, and 39 under 35 U.S.C. § 103 be overturned.

VII. **Conclusion**

In view of the above, Appellants respectfully submit that the rejections of claims 1-39 are overcome. Accordingly, it is respectfully urged that the rejections of claims 1-39 not be sustained.



Stephen R. Tkacs
Reg. No. 46,430
Yee & Associates, P.C.
PO Box 802333
Dallas, TX 75380
(972) 367-2001

APPENDIX OF CLAIMS

The text of the claims involved in the appeal reads:

1. (Original): A method for updating code, the method comprising:

providing an update to a plurality of versions of a program, wherein the program is updated by an installer program and a plurality of versions of the installer program exist;

determining whether a version of the installer program is incorrect with respect to the update;

responsive to the version of the installer program being incorrect, updating the installer program from files in the update; and

installing the update in the program with the updated installer program.

2. (Original): The method as recited in claim 1, wherein a determination that the version of the installer is older than the update indicates that the version of the installer program is incorrect.

3. (Original): The method as recited in claim 1, wherein a determination that the version of the installer is more recent than the update indicates that the version of the installer program is incorrect.

4. (Original): The method as recited in claim 1, wherein the version of the installer program is determined from a single one of a plurality of files contained within the installer program.

5. (Original): The method as recited in claim 1, wherein the updating step comprises:

extracting installer files from the installer program into a directory;

overwriting selected files from the installer program with a corresponding updated file extracted from the update; and

packaging the updated files and remaining installer files into an updated installer program.

6. (Original): The method as recited in claim 5, wherein the packaging step comprises compressing the updated files and remaining installer files to produce an updated installer program.

7. (Original): The method as recited in claim 1, wherein the installer program comprises an install toolkit and the update comprises an update to the install toolkit.

8. (Original): The method as recited in claim 1, wherein the installer program and the update are written in an object-oriented programming language.

9. (Original): The method as recited in claim 1, wherein the installer program comprises a java install toolkit and the update comprises an update to the java install toolkit.

10. (Original): A computer program product in a computer readable media for use in a data processing system for updating code, the computer program product comprising:

first instructions for providing an update to a plurality of versions of a program, wherein the program is updated by an installer program and a plurality of versions of the installer program exist;

second instructions for determining whether a version of the installer program is incorrect with respect to the update;

third instructions, responsive to the version of the installer program being incorrect, for updating the installer program from files in the update; and

fourth instructions for installing the update in the program with the updated installer program.

11. (Original): The computer program product as recited in claim 10, wherein a determination that the version of the installer is older than the update indicates that the version of the installer program is incorrect.

12. (Original): The computer program product as recited in claim 10, wherein a determination that the version of the installer is more recent than the update indicates that the version of the installer program is incorrect.

13. (Original): The computer program product as recited in claim 10, wherein the version of the installer program is determined from a single one of a plurality of files contained within the installer program.

14. (Original): The computer program product as recited in claim 10, wherein the third instructions comprise:

fifth instructions for extracting installer files from the installer program into a directory;

sixth instructions for overwriting selected files from the installer program with a corresponding updated file extracted from the update; and

seventh instructions for packaging the updated files and remaining installer files into an updated installer program.

15. (Original): The computer program product as recited in claim 14, wherein the seventh instructions comprise compressing the updated files and remaining installer files to produce an updated installer program.

16. (Original): The computer program product as recited in claim 10, wherein the installer program comprises an install toolkit and the update comprises an update to the install toolkit.

17. (Original): The computer program product as recited in claim 10, wherein the installer program and the update are written in an object-oriented programming language.

18. (Original): The computer program product as recited in claim 10, wherein the installer program comprises a java install toolkit and the update comprises an update to the java install toolkit.

19. (Original): A system for updating code, the system comprising:

first means for providing an update to a plurality of versions of a program,
wherein the program is updated by an installer program and a plurality of versions of the installer program exist;

second means for determining whether a version of the installer program is incorrect with respect to the update;

third means, responsive to the version of the installer program being incorrect, for updating the installer program from files in the update; and

fourth means for installing the update in the program with the updated installer program.

20. (Original): The system as recited in claim 19, wherein a determination that the version of the installer is older than the update indicates that the version of the installer program is incorrect.

21. (Original): The system as recited in claim 19, wherein a determination that the version of the installer is more recent than the update indicates that the version of the installer program is incorrect.

22. (Original): The system as recited in claim 19, wherein the version of the installer program is determined from a single one of a plurality of files contained within the installer program.

23. (Original): The system as recited in claim 19, wherein the third means comprise:

fifth means for extracting installer files from the installer program into a directory;

sixth means for overwriting selected files from the installer program with a corresponding updated file extracted from the update; and

seventh means for packaging the updated files and remaining installer files into an updated installer program.

24. (Original): The system as recited in claim 23, wherein the seventh means comprise compressing the updated files and remaining installer files to produce an updated installer program.

25. (Original): The system as recited in claim 19, wherein the installer program comprises an install toolkit and the update comprises an update to the install toolkit.

26. (Original): The system as recited in claim 19, wherein the installer program and the update are written in an object-oriented programming language.

27. (Original): The system as recited in claim 19, wherein the installer program comprises a java install toolkit and the update comprises an update to the java install toolkit.

28. (Original): A method in a data processing system for updating code, the method comprising:

receiving an update file for a program, wherein the update file includes an installer to update the program in which a plurality of versions of the installer exist;

determining whether the installer is a newer version of an existing installer for the program;

responsive to the installer being a new version of the existing installer, updating the existing installer to form an updated installer; and

installing code in the update file in the program using the updated installer.

29. (Original): The method as recited in claim 28, wherein the updating step comprises extracting files from the updated installer, overwriting corresponding older files extracted from the existing installer to form updated files, and repackaging the updated files to form the updated installer.

30. (Original): The method as recited in claim 28, wherein the update file, the program, the installer, the existing installer, and the update installer comprise object- oriented code.

31. (Original): The method as recited in claim 30, wherein the object-oriented code is java.

32. (Original): A computer program product in computer readable media for use in a data processing system for updating code, the computer program product comprising:

first instructions for receiving an update file for a program, wherein the update file includes an installer to update the program in which a plurality of versions of the installer exist;

second instructions for determining whether the installer is a newer version of an existing installer for the program;

third instructions, responsive to the installer being a new version of the existing installer, for updating the existing installer to form an updated installer; and

fourth instructions for installing code in the update file in the program using the updated installer.

33. (Original): The computer program product as recited in claim 32, wherein the third instructions comprise extracting files from the updated installer, overwriting corresponding older

files extracted from the existing installer to form updated files, and repackaging the updated files to form the updated installer.

34. (Original): The computer program product as recited in claim 32, wherein the update file, the program, the installer, the existing installer, and the update installer comprise object-oriented code.

35. (Original): The computer program product as recited in claim 34, wherein the object-oriented code is java.

36. (Original): A system for updating code, the system comprising:

first means for receiving an update file for a program, wherein the update file includes an installer to update the program in which a plurality of versions of the installer exist;

second means for determining whether the installer is a newer version of an existing installer for the program;

third means, responsive to the installer being a new version of the existing installer, for updating the existing installer to form an updated installer; and

fourth means for installing code in the update file in the program using the updated installer.

37. (Original): The system as recited in claim 36, wherein the third means comprise extracting files from the updated installer, overwriting corresponding older files extracted from the existing installer to form updated files, and repackaging the updated files to form the updated installer.

38. (Original): The system as recited in claim 36, wherein the update file, the program, the installer, the existing installer, and the update installer comprise object- oriented code.

39. (Original): The system as recited in claim 38, wherein the object-oriented code is java.